Real-Time Rendering of Fast Moving Water

Joshua Glassman*
University of Maryland, Baltimore County

Abstract

In this paper we present a method for modeling and rendering fast moving bodies of water such as rivers in real-time. Rather than use a static polygonal surface, the water surface is represented as a height map on the GPU and is rendered using the recent GPU-based height map rendering algorithms. A simple ray tracer allows fast computation of reflections and refractions. It runs at high frame rates due to effective use of modern graphics hardware, and is intended for use in real-time applications such as video games.

CR Categories: D.1.7 [Software]: Programming Techniques—Visual Programming

Keywords: real-time, water

1 Introduction

Water in real-time computer graphics has been lacking in realism for a long time. Until recently, there were very few games including water surfaces that were anything more than a static surface, with a lot of dynamic texturing. This caused shorelines to appear very unrealistic since the water-to-land transition were static. In addition the water surface had a bizarre appearance when viewed at sharp angles due to the flat representation. Recently, a few games have begun to use deformed surfaces to represent water. A recent paper by Jason L. Mitchell [2004] presents a solution for large bodies of water such as oceans and lakes, but does not solve the problem for fast moving bodies of water such as rivers. We present an extension to Mitchell's solution to account for the vast differences between the appearances of fast moving and large bodies of water.

Our primary contribution is the synthesis and rendering steps used to model fast moving bodies of water in real-time. Our approach allows effects such as reflection and refraction as well as depth effects and directional motion of water, whether through a straight pathway or around a curve.

We will begin by reviewing previous work in the field in section 2, mainly work done in rendering techniques for large bodies of water. After reviewing the various types of techniques and how they relate to ours, we will describe our algorithms in section 3. In section 4, we will present our results and the performance of our system, as well as discuss some possible extensions. We will then conclude with a brief summary and a discussion of possible future directions for this research.

*e-mail: joshums@gmail.com

2 Related Work

Previous work in water synthesis and rendering for computer graphics can be divided into two methods: *simulation* and *approximation* approaches. Additionally, parts of our algorithm rely on *noise* for variety.

2.1 Simulation

In Fournier and Reeves [1986], a simple model was introduced for the modeling and rendering of ocean waves. The technique involved a parametric representation of ocean surfaces, to the extent of modeling interactions with the ocean floor to create curling and breaking, spray, and foam effects.

In Tessendorf [2004] an introduction to the fundamental principles of ocean wave simulation is presented. These techniques are widely used today for off-line rendering, but are not nearly fast enough for real-time applications nor do they consider rendering of fast moving waterways.

General fluid simulation is also a prevalent field of research. Stam introduced a stable model capable of producing complex fluid-like flows in real-time [Stam 1999]. This method made use of classic Navier-Stokes fluid transport formulas with a unique method of representation and solving, optimized for speed over realism. Enright et al. presented a photorealistic approach capable of modeling complex effects such as water being poured into a glass by using sets of positive and negative particles to model the outside and inside of water surfaces [Enright et al. 2002]. Carlson et al. focused on an a system for animating the interplay between existing rigid body and fluid simulations [Carlson et al. 2004]. The system used two-way coupling methods to generate realistic movements for both the fluid and the rigid bodies.

2.2 Approximation

In Hinsinger et al. [2002] was introduced a real-time algorithm for ocean animation, but did not include Fresnel effects nor was the system fast enough for intensive applications such as game engines.

Mitchell animated a Fourier domain spectrum of deep-water ocean waves, then transforming the spectrum into a realistic height map of ocean water via IFFT on the GPU [Mitchell 2004]. A low frequency height map was used to displace the geometry, while low and high frequency normal maps were combined for use in shading. The methods presented allow for compositing of other waveforms such as wakes caused by objects interacting with the water, and for damping of higher frequencies to model the effect of depth variation or the presence of plant matter. Our approach uses many of these techniques, but modifies the algorithms such that they apply to shallow fast moving bodies of water.

Most recently, Hu et al. represented wave geometry view-dependently with a dynamic displacement map with surface detail described by a dynamic bump map [Hu et al. 2006]. The main contributions of this work were the efficient handling of distant geometry by only using a bump map for distant waves, and precomputing a large amount of the information required for rendering. These contributions allow frame rates in a game engine of over 100 on a GeForce 3 graphics card. Our approach makes use of the 'bump

Figure 1: *Simple Geometry.*

map only' approach for distant water surfaces, but the tiling methods are highly modified due to the shape dependent motion of fast moving waterways.

2.3 Noise

Ken Perlin described a method for calculating deterministic noise [Perlin 1985]. Later, this method was revisited and improved in a second paper [Perlin 2002]. A GPU-accelerated version of this algorithm is used to add some variety to various parts of our algorithm.

3 Implementation

In order to have a dynamic water system and to avoid using overly large quantities of graphics memory for texture storage, height fields for the water surface are generated at run-time. In order to do so, we compute a frequency-space representation of the water surface and generate a height field from that via the Inverse Fast Fourier Transform (IFFT), as described in Mitchell [2004]. The generation algorithms used in our method are modified from those presented by Mitchell to account for the unique appearance of fast moving water.

To model view dependent fluctuations of the water surface we displace the surface using a low frequency height map. Lighting calculations are done though a combination of low and high frequency normal maps, using the techniques described in Mitchell [2004]. To keep the algorithm efficient enough for intensive applications, distant water surfaces are not displaced, and instead are only bump mapped as in the approach from Hu et al. [2006].

3.1 Texture Mapping

To account for the highly noticeable directed flow of fast moving water, the textures are squished down so that the water peaks are longer going downstream than across. Additionally, the texture is bent so that the directed flows appear to bend as the waterway bends.

3.2 Interactivity

The most challenging aspect of modeling fast moving water is interactions with the shorelines and other objects. In order to model splashing effects, we use a simple particle system for water droplets. Since the drops move too fast for it to be worth the extra computations, reflection and refraction is skipped and a more simple approach is used for rendering. We instead render the particles as partially transparent billboards, always facing the viewer.

Ripples and wakes are composited onto the water surface using additional height map textures, similar to those used in Mitchell [2004], and additional textures are used to show interactions around stationary objects in the water surface. However, since the water is moving quickly, the composited textures need to be distorted over time in the direction of water flow. For this effect hardware accelerated Perlin noise is used to both distort and deteriorate the effects of the ripples.

3.3 Rendering

In order to render the water surface with correct lighting, we first need to generate a valid normal map. The normal map is approximated by calculating the differences between adjacent height values at each point in the height map, and taking the cross product of those differences. This produces a normal map which is locally approximated with accuracy dependent on the resolution of the height map.

4 Results

At this point in time, we currently have a system which displaces mesh geometry and renders with Fresnel reflection and refraction. Each frame the system takes as input a height map, generates an approximate normal map, and uses the two in the final rendering algorithm. Reflection and refraction colors are currently retrieved from an arbitrary cube map texture, though in a dynamic environment reflection and refraction textures could be generated each frame. Remaining is the implementation of one of a number of fast water height map generation algorithms, with some small modifications

Figure 2: *Without Displacement.*

to approximate the differences between large bodies of water and fast moving waterways. The best method to use for this is probably the one used by Hu et al. [2006] since the water would be moving fast enough that it is unlikely that cyclic repetitions would be noticeable, especially due to the use of noise to augment the appearance at the shorelines and other interactive locations. Although this method takes up more texture memory due to the needed storage of several textures, the increased rendering speed is very attractive and will allow the system to be used across more systems. However, if this were to be used on a device with tight memory constraints such as a console game system, the method presented by Mitchell [2004] would likely be more appropriate.

Figure 1 shows an image resulting from a combination of all of our techniques. This scene was rendered at xx frames per second on a GeForce 7800 graphics card.

Figure 2 shows an image rendered using our techniques but without displacing the original geometry. This scene was rendered at xx frames per second on a GeForce 7800 graphics card.

5 Future Directions

We have presented an approach to synthesizing and rendering fast moving water entirely on the graphics processor. Our approach uses modifications of known methods to generate both low frequency height maps for surface displacement and broad spectrum normal maps for lighting calculations. This allows for compositing of additional height maps to model the effects of objects interacting with the water surface. We have also described a rendering method which approximates the appearance of fast moving waterways, which includes reflection and refraction. We concluded by demonstrating the results and performance of our system on modern graphics hardware.

While a number of papers have discussed generating physically accurate or good approximations to the appearance of ocean water, fast moving waterways have not been discussed. An algorithm which generates a better representation of fast moving water would be an excellent extension to the techniques presented here.

References

ALEX VLACHOS, J. R. I., AND OAT, C. 2002. Rippling reflective and refractive water. In *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks*, W. Engel, Ed. Wordware, Plano, Texas.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, ACM Press, New York, NY, USA, 377–384.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 736–744.

FOURNIER, A., AND REEVES, W. T. 1986. A simple model of ocean waves. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 75–84.

GAMITO, M. N., AND MUSGRAVE, F. K. 2002. An accurate model of wave refraction over shallow water. *Computers & Graphics 26*, 2, 291–307.

HINSINGER, D., NEYRET, F., AND CANI, M.-P. 2002. Interactive animation of ocean waves. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, New York, NY, USA, 161–166.

HU, Y., VELHO, L., TONG, X., GUO, B., AND SHUM, H. 2006. Realistic, real-time rendering of ocean waves: Research articles. *Comput. Animat. Virtual Worlds 17*, 1, 59–67.

JOHN R. ISIDORO, A. V., AND BRENNAN, C. 2002. Rendering ocean water. In *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks*, W. Engel, Ed. Wordware, Plano, Texas.

KIM, J., CHA, D., CHANG, B., KOO, B., AND IHM, I. 2006. Practical animation of turbulent splashing water. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 335–344.

KIPFER, P., SEGAL, M., AND WESTERMANN, R. 2004. Uberflow: a gpu-based particle engine. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, ACM Press, New York, NY, USA, 115–122.

LOVISCACH, J. 2003. Complex water effects at interactive frame rates. In *WSCG*.

MASTIN, G. A., WATTERBERG, P. A., AND MAREDA, J. F. 1987. Fourier synthesis of ocean scenes. *IEEE Comput. Graph. Appl. 7*, 3, 16–23.

MAX, N. L. 1981. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 317–324.

MITCHELL, J. L. 2004. Real-time synthesis and rendering of ocean water. In *ATI Research Technical Report*. Marlboro, MA.

MORELAND, K., AND ANGEL, E. 2003. The fft on a gpu. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 112–119.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 287–296.

PERLIN, K. 2002. Improving noise. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 681–682.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 335–342.

PREMOZE, S., AND ASHIKHMIN, M. 2001. Rendering natural waters. *Computer Graphics Forum 20*, 4, ??–??

REEVES, W. T. 1983. Particle systems–a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2, 91–108.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 343–352.

SCHNEIDER, J., AND WESTERMANN, R. 2001. Towards real-time visual simulation of water surfaces. In *VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001*, Aka GmbH, 211–218.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 910–914.

STAM, J. 1999. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 121–128.

SU, W. Y., AND HART, J. C. 2005. A programmable particle system framework for shape modeling. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, ACM Press, New York, NY, USA, 277.

TESSENDORF, J. 2004. Simulating ocean water.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 965–972.